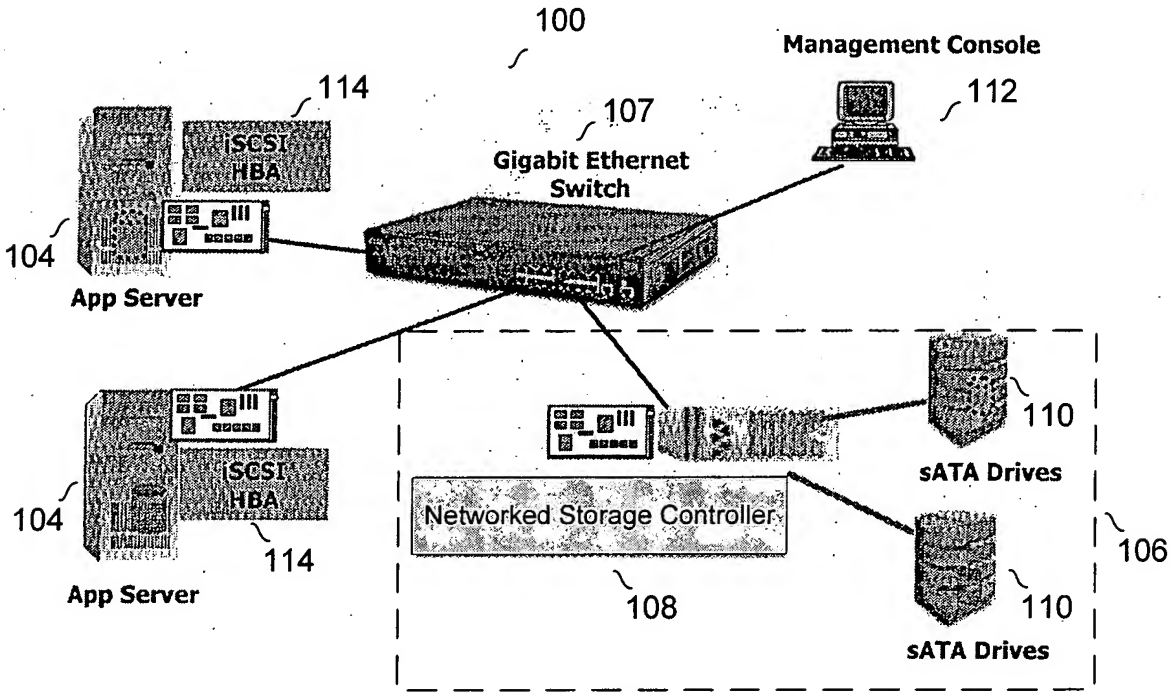


FIGURE 1

BEST AVAILABLE COPY

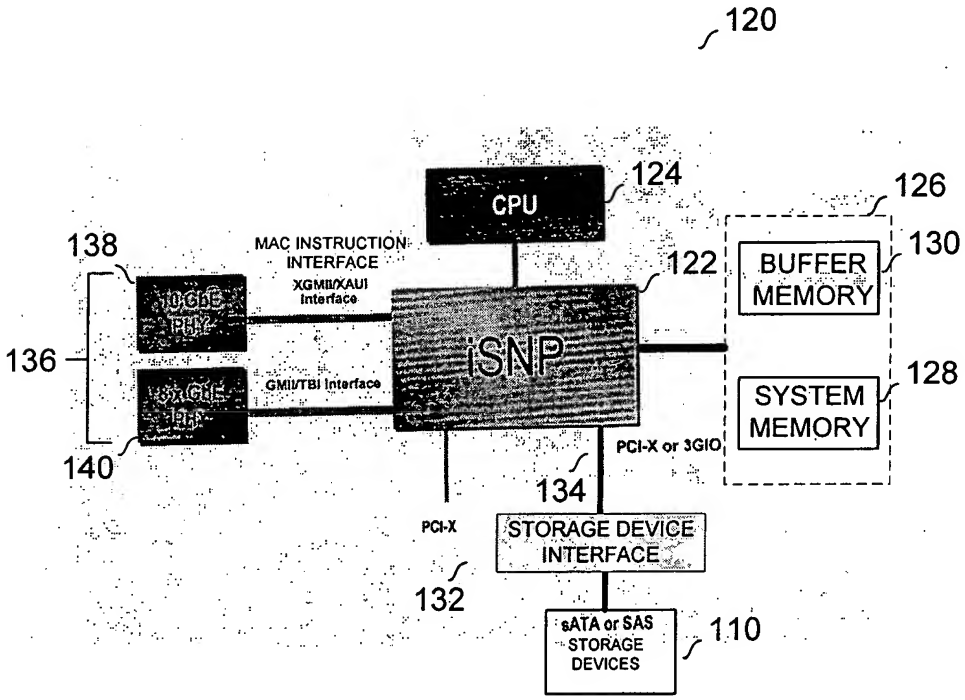

120
CPU
124
126
138
MAC INSTRUCTION INTERFACE
XGMII/XAUI Interface
122
BUFFER MEMORY
130
10 GbE PHY
136
iSNP
GMII/TBI Interface
1 GbE PHY
SYSTEM MEMORY
128
140
PCI-X or 3GIO
134
PCI-X
STORAGE DEVICE INTERFACE
132
sATA or SAS STORAGE DEVICES
110

FIGURE 2

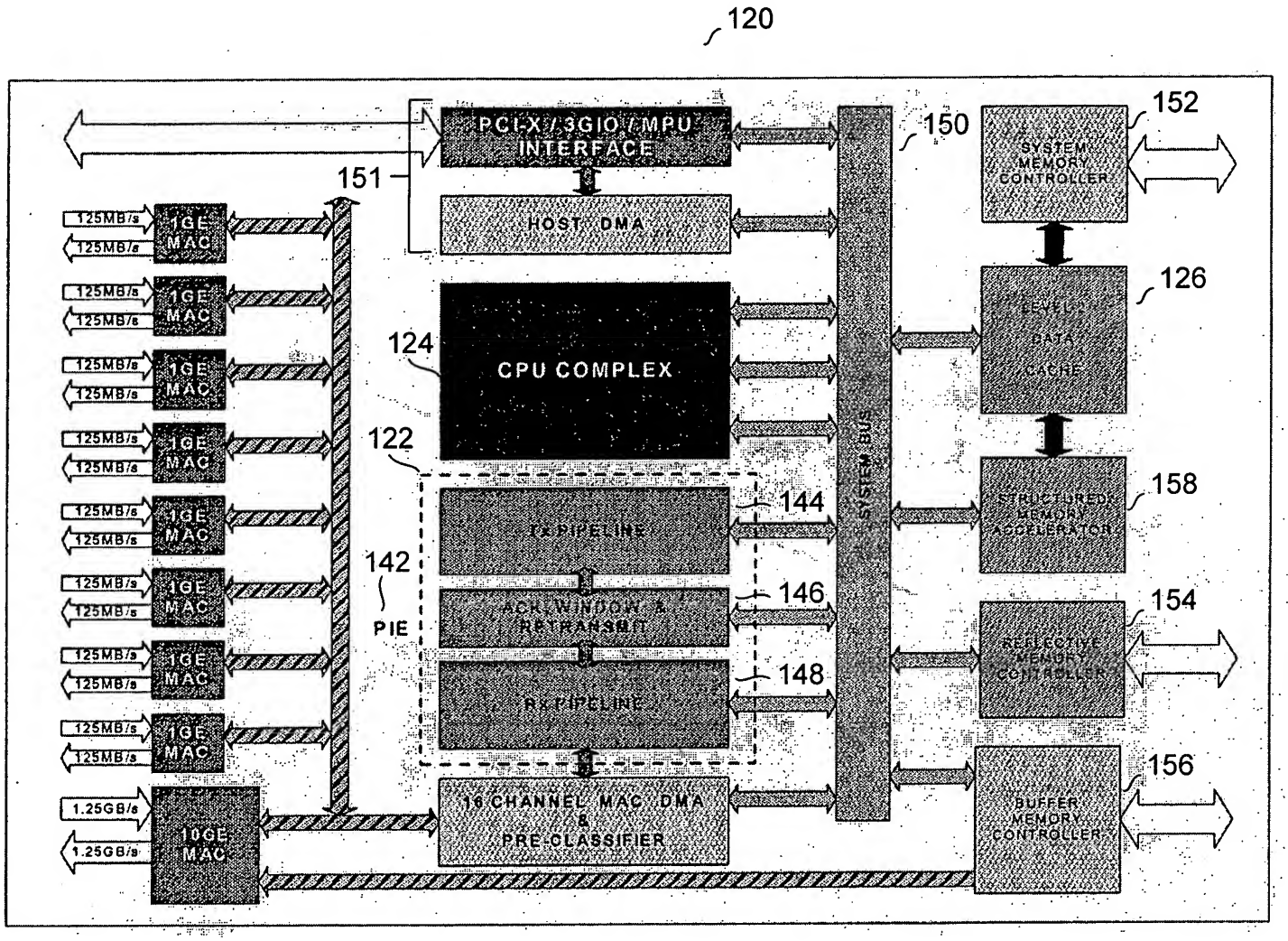


FIGURE 3A

BEST AVAILABLE COPY

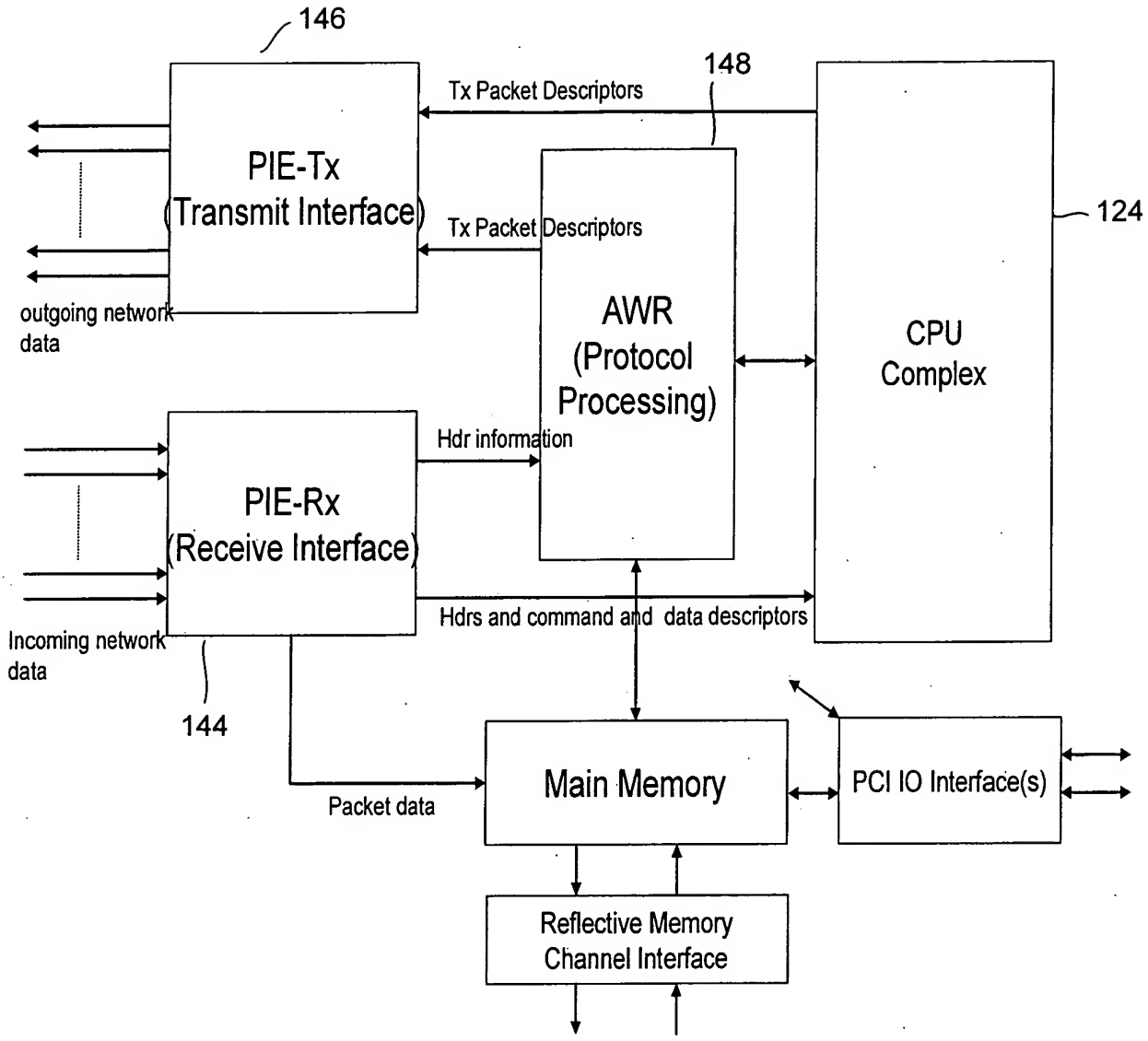

146
148
Tx Packet Descriptors
PIE-Tx
(Transmit Interface)
Tx Packet Descriptors
124
AWR
(Protocol
Processing)
CPU
Complex
outgoing network
data
Hdr information
PIE-Rx
(Receive Interface)
Hdrs and command and data descriptors
Incoming network
data
144
Main Memory
PCI IO Interface(s)
Packet data
Reflective Memory
Channel Interface

FIGURE 3B

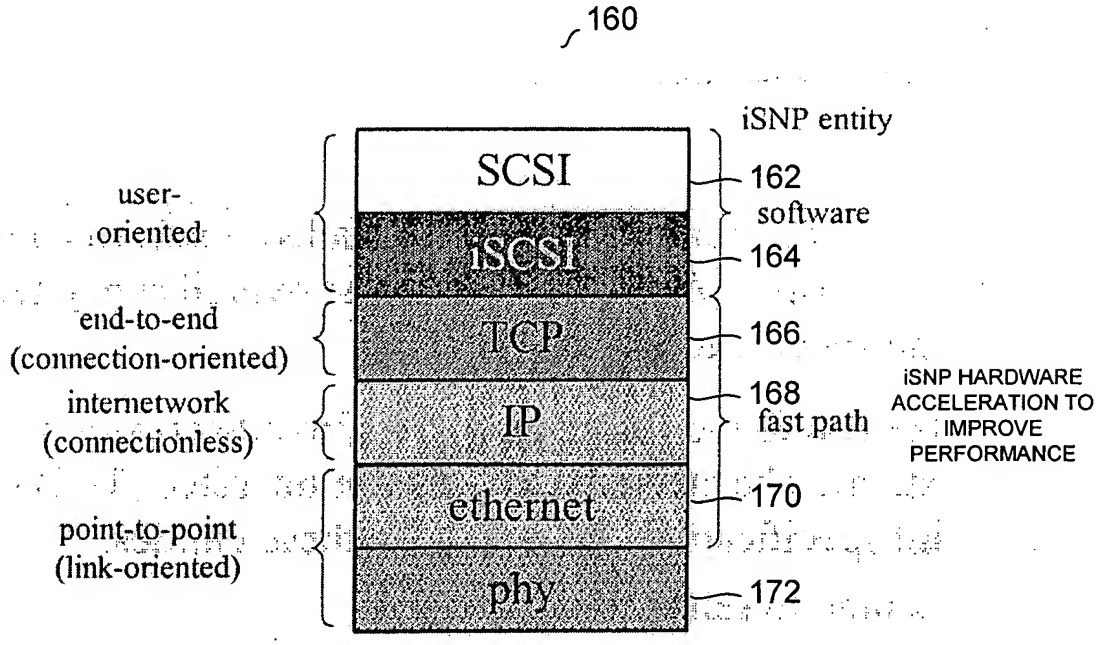


FIGURE 4A

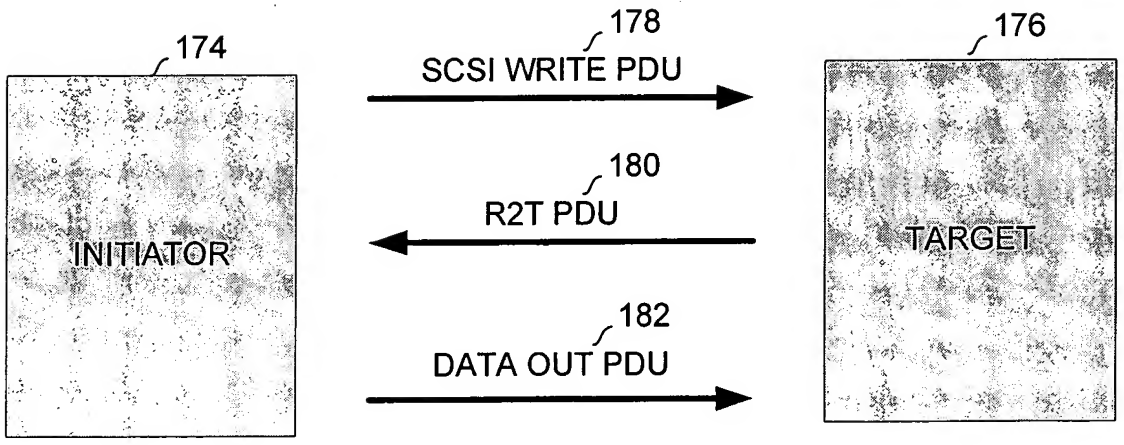


FIGURE 4B

segment	iSCSI PDU			
BHS (basic header segment, bytes 0-47)	B0[6]=I (immed delivery) B0[5:0]=opcode, initiator / target 'h01=SCSI command / 'h21=SCSI response 'h02=SCSI task mgmt req / 'h22=SCSI task mgmt rsp 'h03=login command / 'h23=login response 'h05=SCSI data out / 'h25=SCSI data in 'h06=logout command / 'h26=logout response 'h10=SNACK (seq# ack) req / 'h31=R2T (rdy to xfr) 'h3f=reject		B1-B3=opcode specific fields SCSI command: B1[7]=F (final), B1[6]=R (read), B1[5]=W (write) B1[2:0]=task attrib (0=untagged, 1=simple, 2=ord, 3=hoq, 4=aca) B3=CRN (command reference number) SCSI data out: B1[7]=F SCSI data in: B1[7]=F, B1[2]=O (overflow), B1[1]=U (underflow), B1[0]=S (status, if set: B3=status, B20-23=residual count, B24-27=statusSN) SCSI response: B1[2]=O, B1[1]=U, B1[4]=O for bidir read, B1[3]=U bi rd B2=response (0=cmd complete, 1=targ fail).	
	B4=total AHS length		B5-7=data segment length (not including pad)	
	B8-15=logical unit number (LUN) (64 bits)			
	B16-19=initiator task tag			
	B20-47=opcode specific fields (7 words) SCSI command: B20-23=expected data length, B24-27=cmdSN (cmd seq number), B28-31=expStatSN (acks status thru SN-1) B32-47=CDB (16 bytes) R2T: B20-23=targ transfer tag, B24-27=statSN (for next status), B28-31=expCmdSN (cmd ack to init), B32-35=maxCmdSN B36-39=R2TSN (R2T PDU number, starts at 0), B40-43=buffer offset, B44-47=desired data length (in bytes) SCSI data out: B20-23=targ transfer tag from R2T (or F's), B28-31=expStatSN (expected status sequence number) B36-39=dataSN, B40-43=buffer offset (for this payload relative to complete data transfer) SCSI data in: B20-23=residual count, B24-27=statSN, B28-31=expCmdSN, B32-35=maxCmdSN, B36-39=dataSN, B40-43=buffer offset SCSI response: B24-27=statSN, B28-31=expCmdSN, B32-35=maxCmdSN, B36-39=expDataSN, B40-43=bidirectional read residual count, B44-47=residual count			
AHS (additional header seg, optional)	B0=B1=AHS length		B2=AHS type B2[7]=drop B2[5:0]=ahs code 1=extended CDB 2=exp bidir read length	B3=AHS specific
	extended CDB: B4..n=extended CDB (if necessary pad to full word) bidir read length: B4=expected read data length			
hdr digest (optional)	CRC for header segment(s)			
data seg (optional)	Data (if necessary pad to full word)			
data digest (optional	CRC for data segment			

FIGURE 5A

word#	TCP Segment Header Fields			
0	source port[15:0]			destination port[15:0] 'h0015=FTP, 17=TELNET, 19=SMTP, 1D8=HTTP, CBC=iSCSI
1	sequence number[31:0]			
2	acknowledgement number[31:0] (seq# of next expected octet, acks all previous octets)			
3	hdr length[3:0] (in words)	reserved[5:0]	flags[5:0] bit5=urgent ptr valid bit4=ack number valid bit3=PSH bit2=RST (reset connection) bit1=SYN (for connect/close) bit0=FIN (for close)	window[15:0] (number of octets the receiver is able to receive)
4	checksum[15:0] (same as IP, but covers header and data)			urgent pointer[15:0]
optional	options (if any) plus padding (variable length depending on the number and type of options)			
	data			

FIGURE 5B

word#	IP Packet Header Fields				
0	version[3:0]	IHL[3:0] (header len in words, min 5=no options)	TOS[7:0] (type of service, specifies precedence, delay, throughput, and reliability parameters)	TLEN[15:0] (total length including header, in octets)	
1	ID[15:0] (with src addr, dst addr, and user protocol uniquely identifies the IP datagram)			flag[2:0] flag[1]=dont frag flag[0]=more frags	fragment offset[12:0] (in 64-bit units)
2	TTL[7:0] (time to live, decremented at each hop, if 0 discard the datagram)		Protocol[7:0] 1=ICMP 6=TCP 17=UDP	Header checksum [15:0] (1's complement of the 1's complement sum of data interpreted 16 bits at a time, with end-around carry)	
3	Source IP address[31:0] (coded to allow a variable number of bits to specify the network and station)				
4	Destination IP address[31:0] (as for source address, 224-239.x.x.x=multicast)				
:	Options (if any) plus padding (variable length depending on the number and type of options)				
:	Data (multiple of 8 bits in length)				

FIGURE 5C

ethernet frame field	pre-amble	SFD (start frame)	DA (dst addr)	SA (src addr)	type (ethernet II) or length of info (IEEE 802.3)	information	FCS (frame chk seq)	extension
#octets	7	1	6	6	2	46-1500	4	
notes					'h0800=IP 'h0806=ARP 'h86dd=IPv6 'h8100=VLAN (insert w/2-byte tag before type/length) <=>'h05dc=length (rfc1042, insert w/oonstant 0xaaaa03_000000 before type)	jumbo: max 9000 GE: min=512 for half duplex CSMA/CD) <min: add pad bytes		special nondata symbols for half duplex CSMA/CD

FIGURE 5D

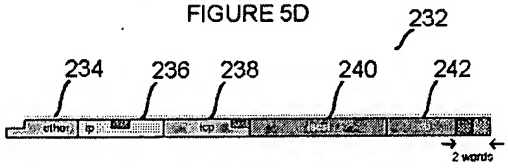


FIGURE 5E

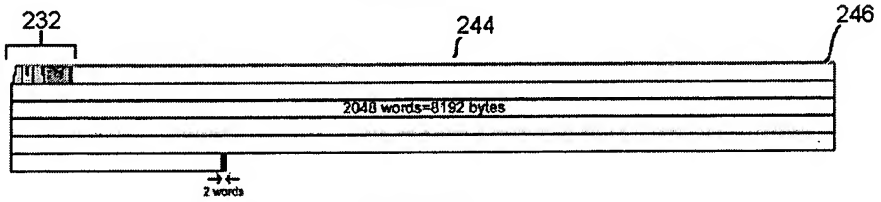


FIGURE 5F

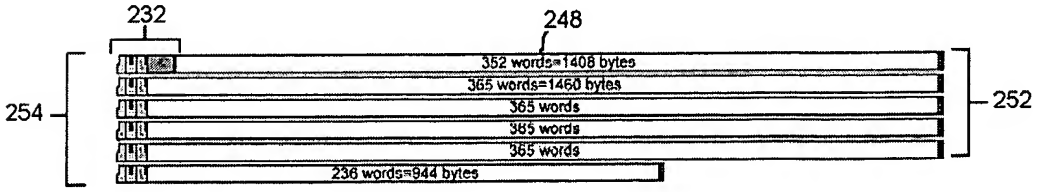


FIGURE 5G

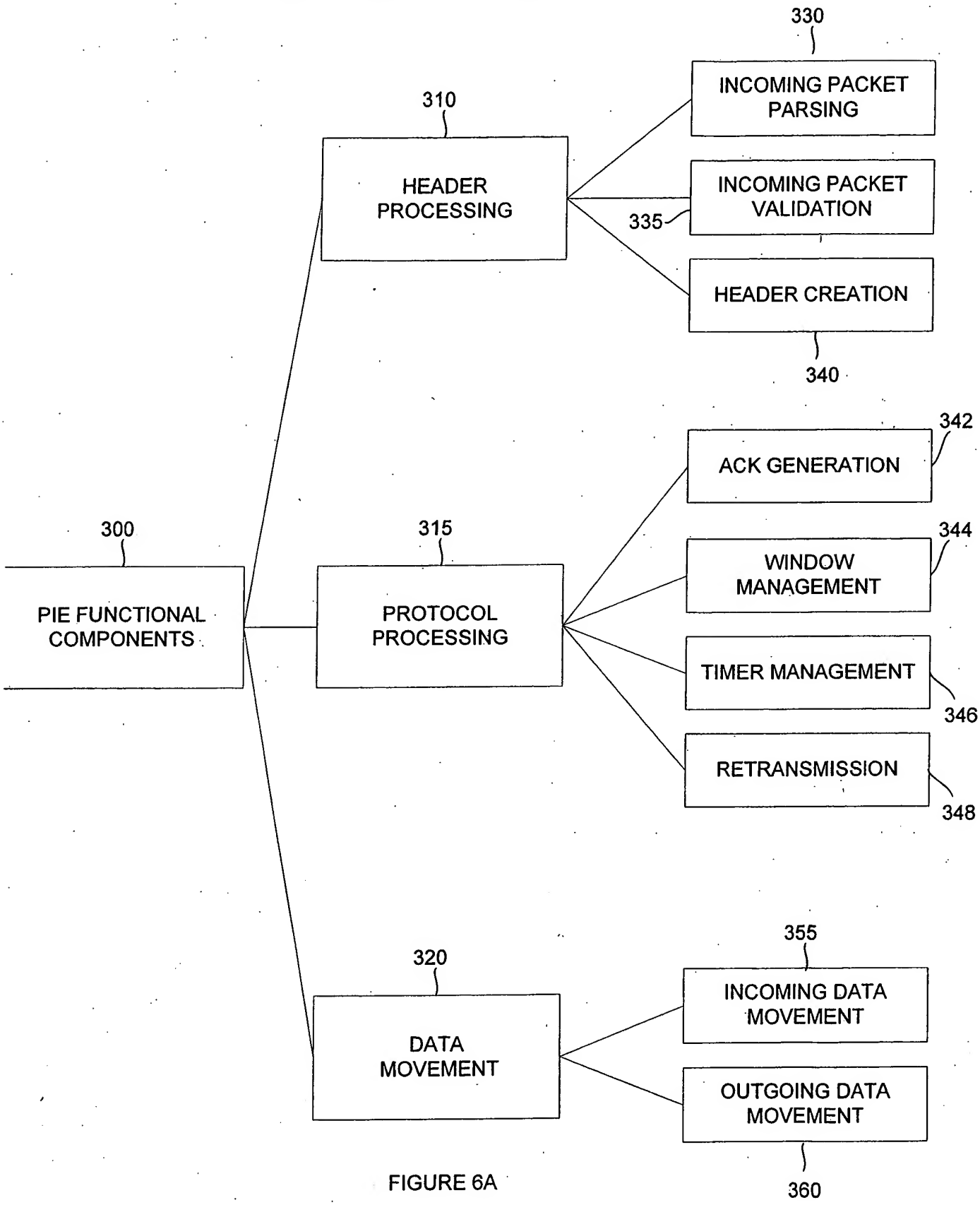

300
PIE FUNCTIONAL COMPONENTS
310
HEADER PROCESSING
315
PROTOCOL PROCESSING
320
DATA MOVEMENT
330
INCOMING PACKET PARSING
335
INCOMING PACKET VALIDATION
340
HEADER CREATION
342
ACK GENERATION
344
WINDOW MANAGEMENT
346
TIMER MANAGEMENT
348
RETRANSMISSION
355
INCOMING DATA MOVEMENT
360
OUTGOING DATA MOVEMENT

FIGURE 6A

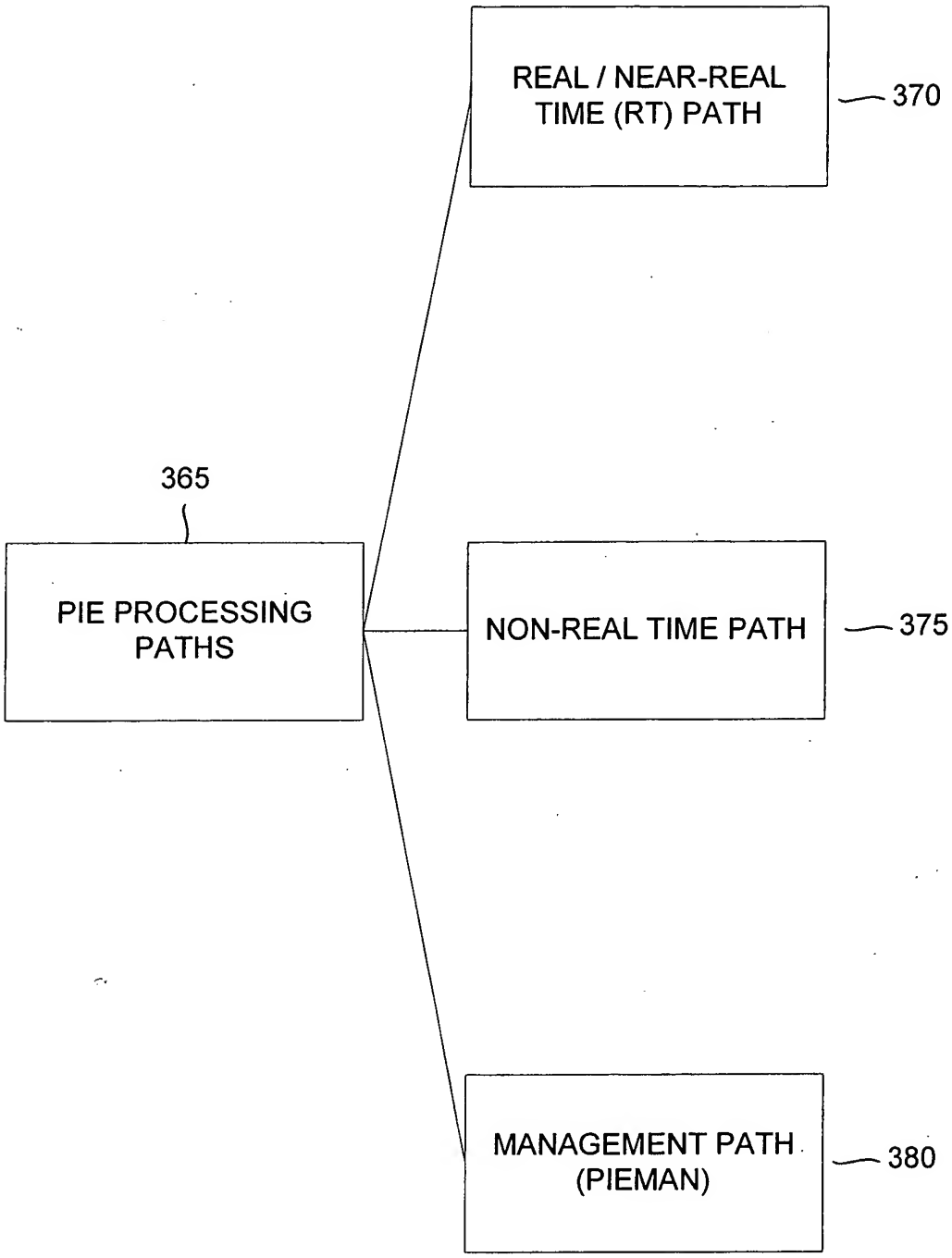

REAL / NEAR-REAL TIME (RT) PATH
370
365
PIE PROCESSING PATHS
NON-REAL TIME PATH
375
MANAGEMENT PATH (PIEMAN)
380

FIGURE 6B

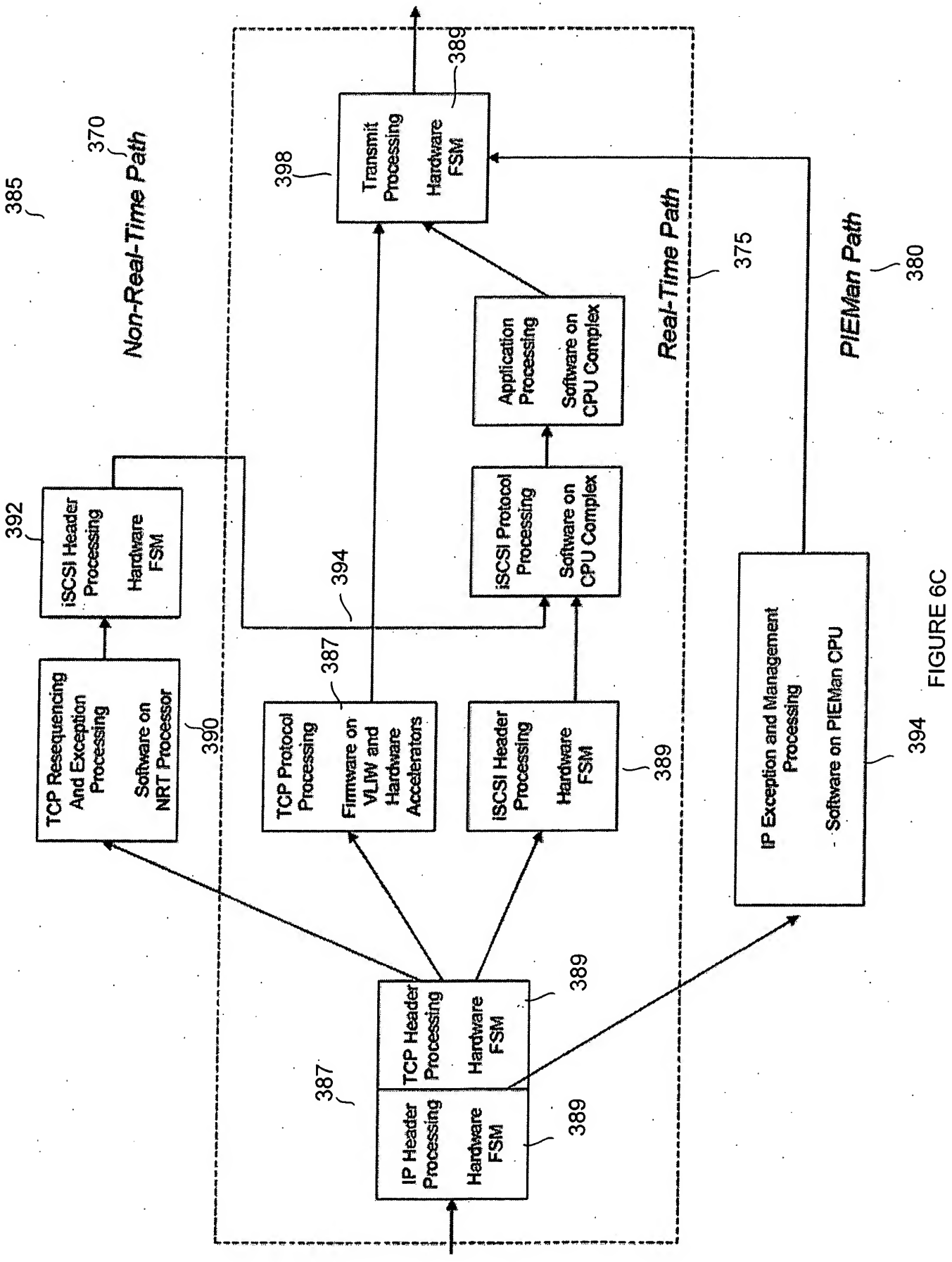
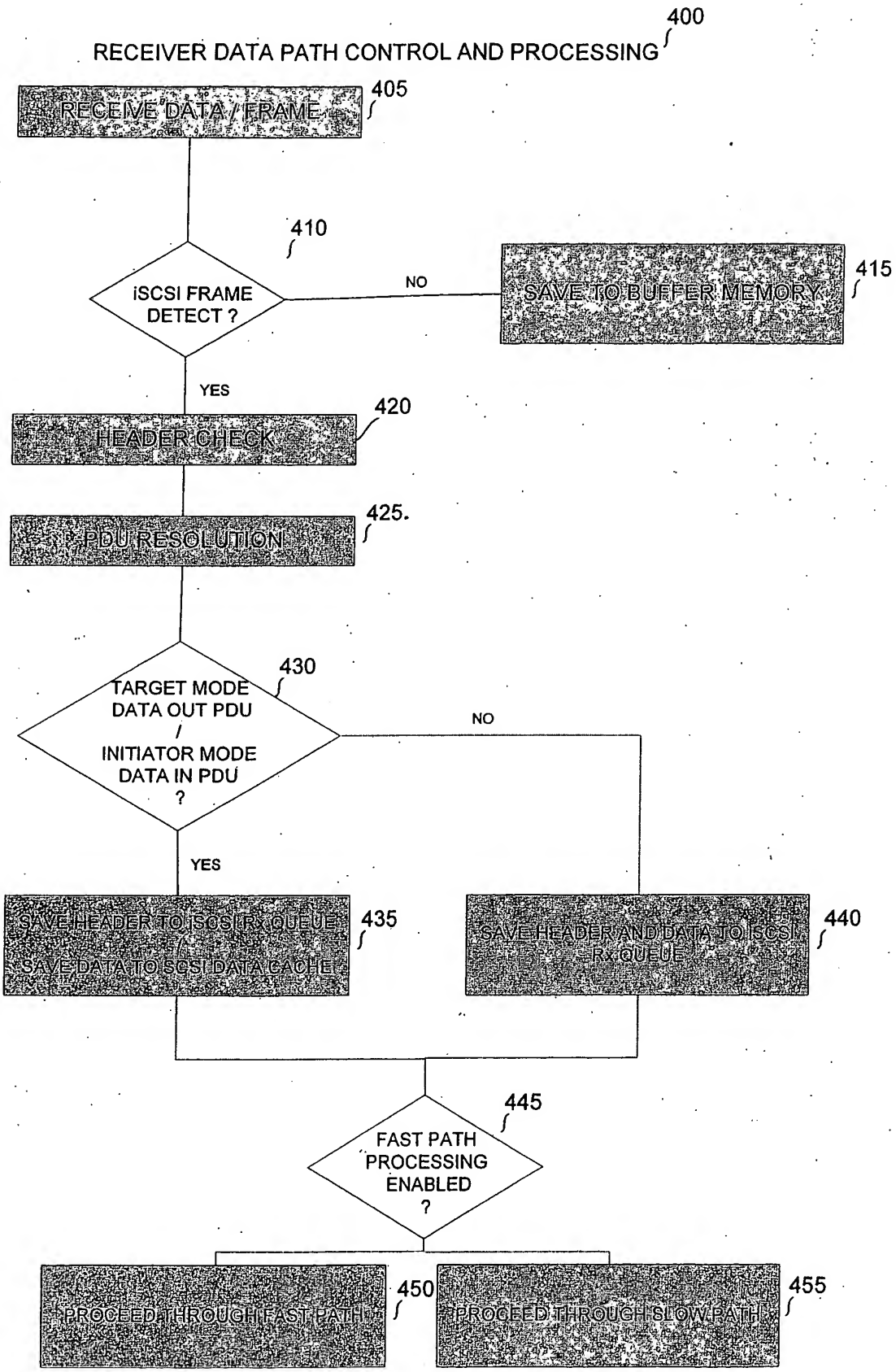

385
370
Non-Real-Time Path
375
Real-Time Path
380
PIEMan Path
398
Transmit Processing
Hardware FSM
389
392
iSCSI Header Processing
Hardware FSM
394
Application Processing
Software on CPU Complex
iSCSI Protocol Processing
Software on CPU Complex
TCP Resequencing And Exception Processing
Software on NRT Processor
390
387
TCP Protocol Processing
Firmware on VLIW and Hardware Accelerators
iSCSI Header Processing
Hardware FSM
389
IP Exception and Management Processing
Software on PIEMan CPU
394
387
TCP Header Processing
Hardware FSM
389
IP Header Processing
Hardware FSM
389

FIGURE 6C

FIGURE 7


400
RECEIVER DATA PATH CONTROL AND PROCESSING
RECEIVE DATA / FRAME
405
iSCSI FRAME DETECT ?
410
NO
SAVE TO BUFFER MEMORY
415
YES
HEADER CHECK
420
PDU RESOLUTION
425
TARGET MODE DATA OUT PDU / INITIATOR MODE DATA IN PDU ?
430
NO
YES
SAVE HEADER TO iSCSI Rx QUEUE
SAVE DATA TO SCSI DATA CACHE
435
SAVE HEADER AND DATA TO iSCSI Rx QUEUE
440
FAST PATH PROCESSING ENABLED ?
445
PROCEED THROUGH FAST PATH
450
PROCEED THROUGH SLOW PATH
455

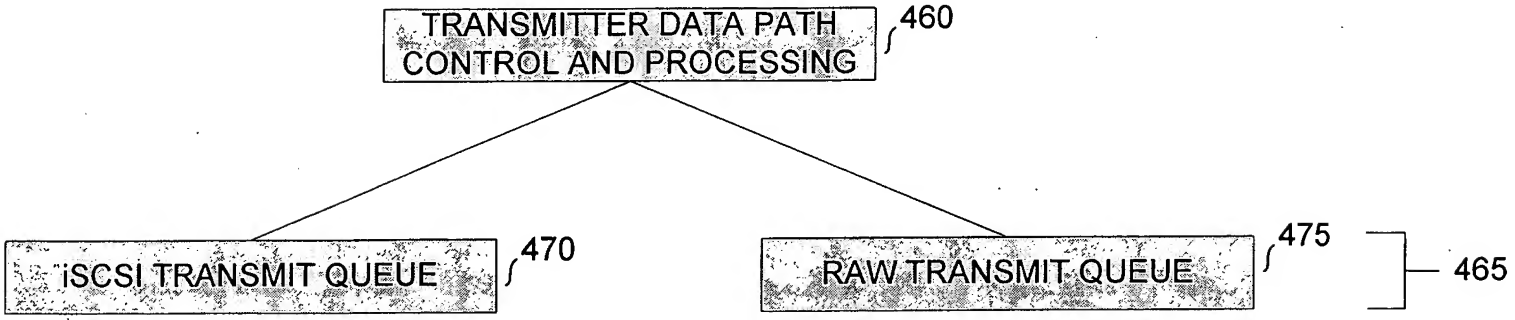


FIGURE 8A

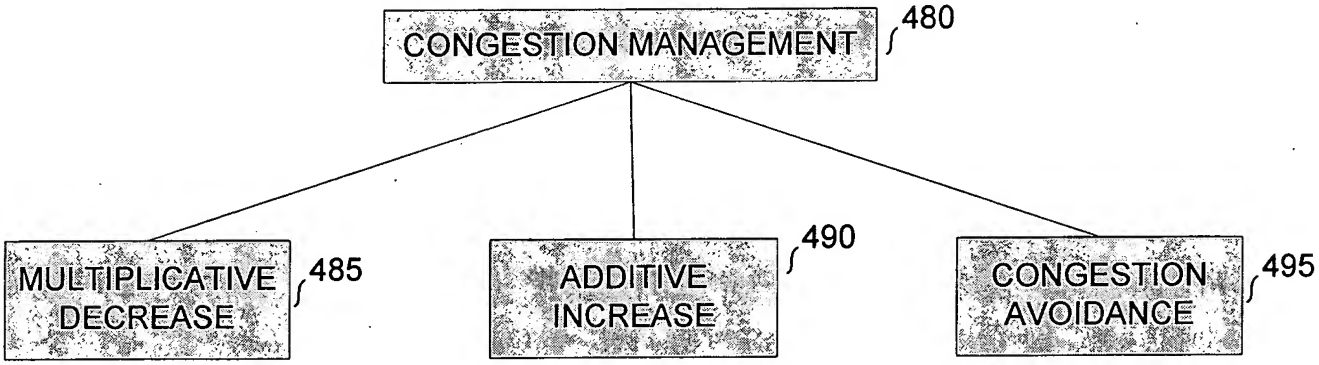


FIGURE 8B

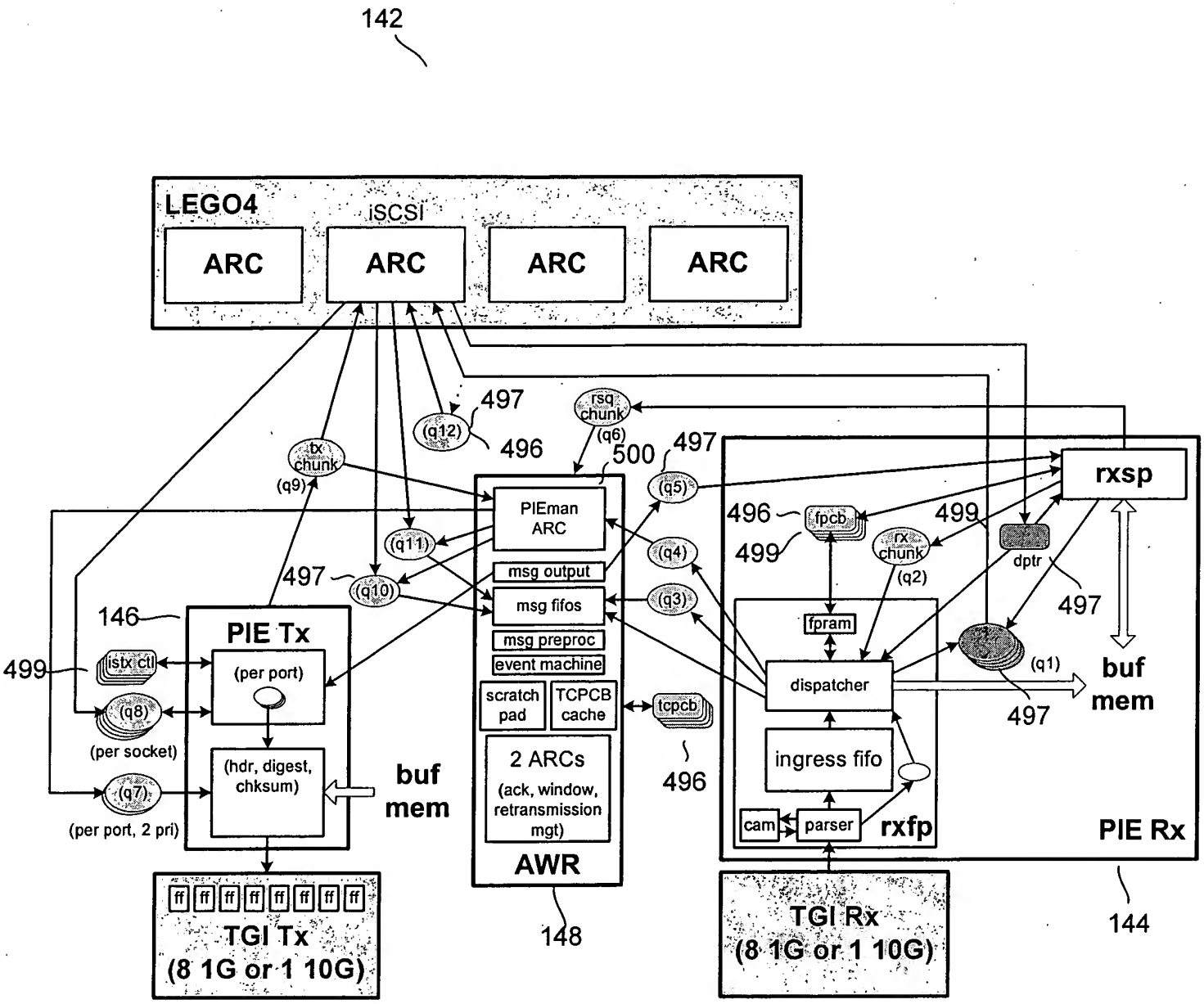

142
LEGO4
iSCSI
ARC
ARC
ARC
ARC
497
(q12)
496
rsq chunk
(q6)
500
497
tx chunk
(q9)
(q5)
PIEman ARC
(q11)
(q4)
msg output
497
(q10)
msg fifos
(q3)
146
PIE Tx
msg preproc
event machine
499
istx ctl
(per port)
scratch pad
TCPCB cache
tcpcb
(q8)
(per socket)
2 ARCs
(ack, window, retransmission mgt)
(q7)
(per port, 2 pri)
(hdr, digest, chksum)
buf mem
496
AWR
148
TGI Tx
(8 1G or 1 10G)
rxsp
496
fpcb
499
499
rx chunk
(q2)
dptr
497
fpram
(q1)
buf mem
dispatcher
497
ingress fifo
cam
parser
rxfp
PIE Rx
144
TGI Rx
(8 1G or 1 10G)

FIGURE 9

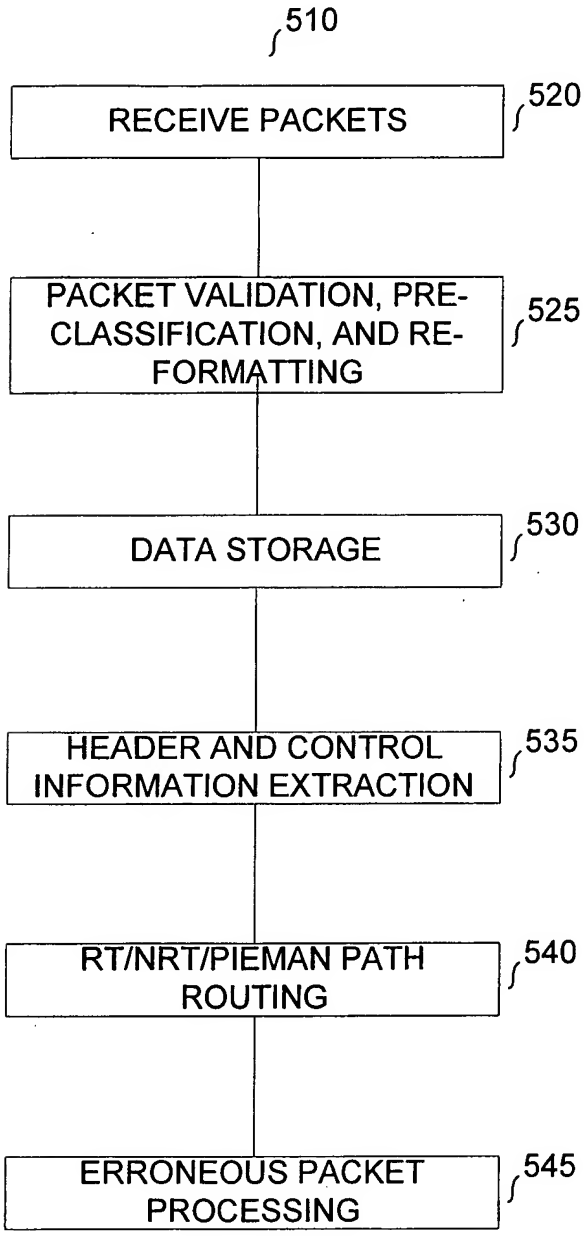

510
RECEIVE PACKETS
520
PACKET VALIDATION, PRE-CLASSIFICATION, AND RE-FORMATTING
525
DATA STORAGE
530
HEADER AND CONTROL INFORMATION EXTRACTION
535
RT/NRT/PIEMAN PATH ROUTING
540
ERRONEOUS PACKET PROCESSING
545

FIGURE 10

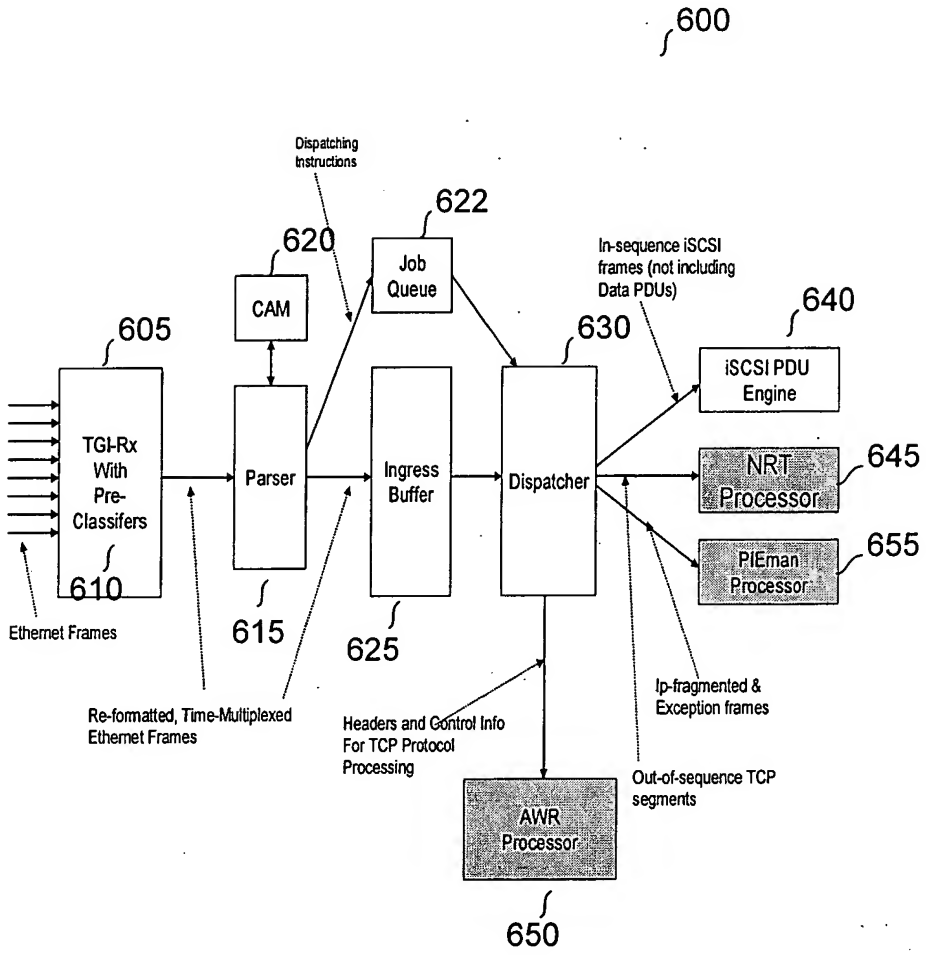

600
Dispatching Instructions
622
620
Job Queue
CAM
In-sequence iSCSI frames (not including Data PDUs)
640
605
630
iSCSI PDU Engine
TGI-Rx With Pre-Classifers
Parser
Ingress Buffer
Dispatcher
NRT Processor
645
PIEman Processor
655
610
615
625
Ethernet Frames
Re-formatted, Time-Multiplexed Ethernet Frames
Headers and Control Info For TCP Protocol Processing
Ip-fragmented & Exception frames
Out-of-sequence TCP segments
AWR Processor
650

FIGURE 11

TGI field	Description
tag[3:0]	Indicates the type of information in the dword, as preclassified by the TGI 0: invalid (interframe) 1: E (ethernet header) 2: EV (ethernet header, VLAN present) 3: E8 (ethernet header, 802.3 rfc1042 format) 4: EV8 (ethernet header, 802.3 rfc1042 format, VLAN present) 5: I (IP) 6: IO (IP options) 7: IF (IP fragmented) 8: T (TCP) 9: TO (TCP option) 10: U (UDP) 11: spare 12: S (iSCSI, i.e. TCP and DPORT or SPORT matches iSCSI) 13: O (other, e.g. not IP, not TCP or UDP) 14: G (good EOF, dword holds checksum, frame length) 15: B (bad EOF, assert early if checksum or other error detected)
off[2:0]	Indicates the byte offset to tag boundary within dword, 0=left side

FIGURE 12A

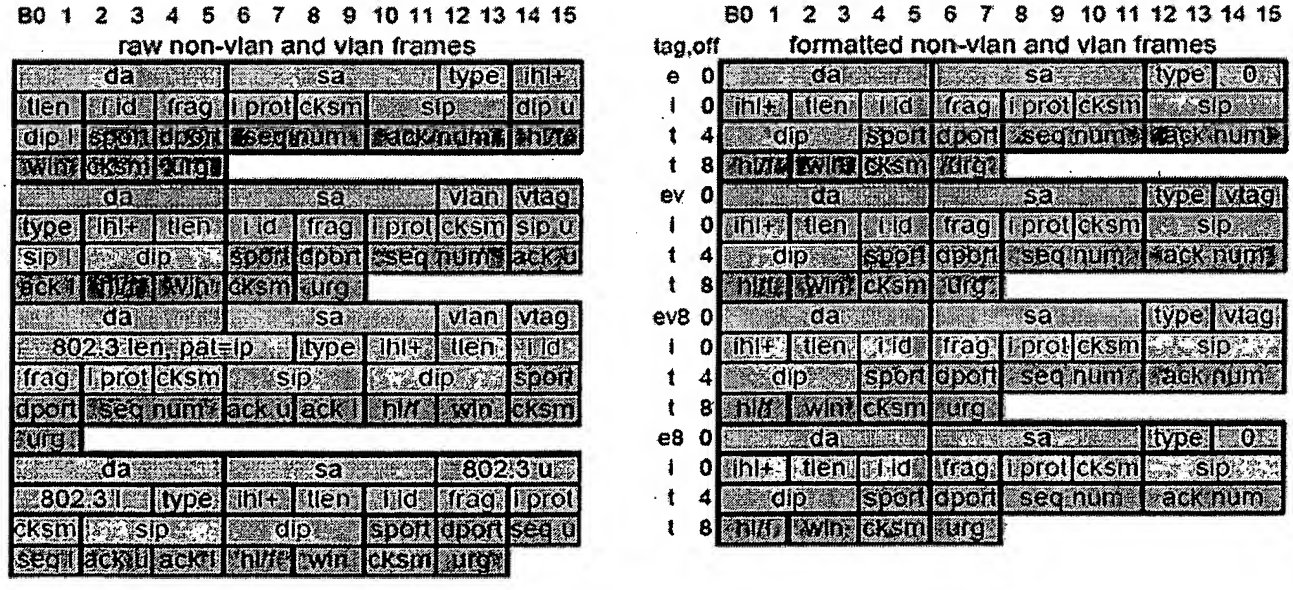


FIGURE 12B

#bits	Field	Description
4	State	State machine state
2	ts_offst	Timestamp offset (0=none, 1=22B from start of TCP header, 2=23B, 3=24B)
1	t_left	TCP header starts in left half of dword
4	Reason	If nonzero, the reason from slow-path processing
1	Msw_par	Job FIFO MS word parity
1	d_left	1st dword stored (at next dword write qword to ingress fifo)
64	d_data	Stored dword
2	d_par	Stored dword parity
80		Total bits

FIGURE 13A

Slow-Path Reason codes (*=set by dispatcher):

0 : nop -- fastpath iSCSI
1: ARP frame
2: other non-IP (not ARP) frame
3: IP fragment (if not, fragment zero could be iSCSI)
4: TCP but not iSCSI or runt iSCSI (flen<0x38)
5: UDP frame
6: ICMP frame
7: other IP frame (not IP fragment, TCP, UDP, or ICMP)
8: iSCSI, IP fragment zero
9: iSCSI, no socket ID found in CAM
a: iSCSI, unsupported option
*b: iSCSI, fastpath disabled
*c: iSCSI, out of sequence
*d: iSCSI, bad data boundary

FIGURE 13B

CAM_LOAD I, data = Write CAM entry I with specified data; Set Valid bit;

CAM_READ I = Read data contained in CAM entry I;

CAM_INV I = Clear the valid bit for CAM entry I;

CAM_REQ P = Initiate CAM search with Key elements written in the CAM-Key register for network port P;

CAM_RESULT P = Fetch result from CAM search for network port P;

FIGURE 14

#bits	Field	Description
2	TS offset	If nonzero, byte offset minus 1 from start of TCP options to timestamp field
4	TCP option length	Size in words of TCP options
4	Slow-path Reason Code	If nonzero, packet takes slow path. Reason codes (*=set by dispatcher): 0: nop, fastpath iSCSI 1: ARP 2: other non-IP, non-ARP frame 3: IP fragment 4: TCP (not iSCSI) or runt iSCSI (flen<0x38) 5: UDP 6: ICMP 7: other IP (not IP fragment, TCP, UDP, or ICMP) 8: iSCSI, IP fragment zero 9: iSCSI, no socket ID a: iSCSI, unsupported option *b: iSCSI, fastpath disabled *c: iSCSI, out of sequence *d: iSCSI, bad data boundary
1	ID valid	iSCSI socket ID valid
1	Init	Initiator mode
4	IP option length	Size in words of IP options
10	Socket ID	iSCSI socket ID
1	VLAN	16th byte contains VLAN tag (IP frame)
1	802.3	802.3 rfc1042 coding was removed from ethernet header (IP frame)
14	frame length	Length of formatted frame in bytes
16	partial checksum	Checksum for UDP or partial TCP segment (info for PIEman)

FIGURE 15

#bits	field	description	notes	
58	job	jff output		
13	roffst	iff read offset	for random access of iff	
13	rckpt	iff read checkpoint (start of frame)	for calculating iff discard point (add flength)	
10	fctr	frame qword counter		
32	seq	TCP sequence number	if iSCSI (except flags)	for msgRxNotify and msgRxFrame, also need some job fields
32	ack	TCP acknowledgement number		
32	flgs	flags, TCP flags, TCP window size		
32	ts	TCP timestamp		
32	ets	TCP echo timestamp		
288	rcp0-rcp8	Rx chunk pointer 0 up to 8	depending on flength	for msgRxFrame
546		total (approx 69 bytes per port, total 552B)		

Figure 16 . Dispatcher Per-port Frame Context

FIGURE 16

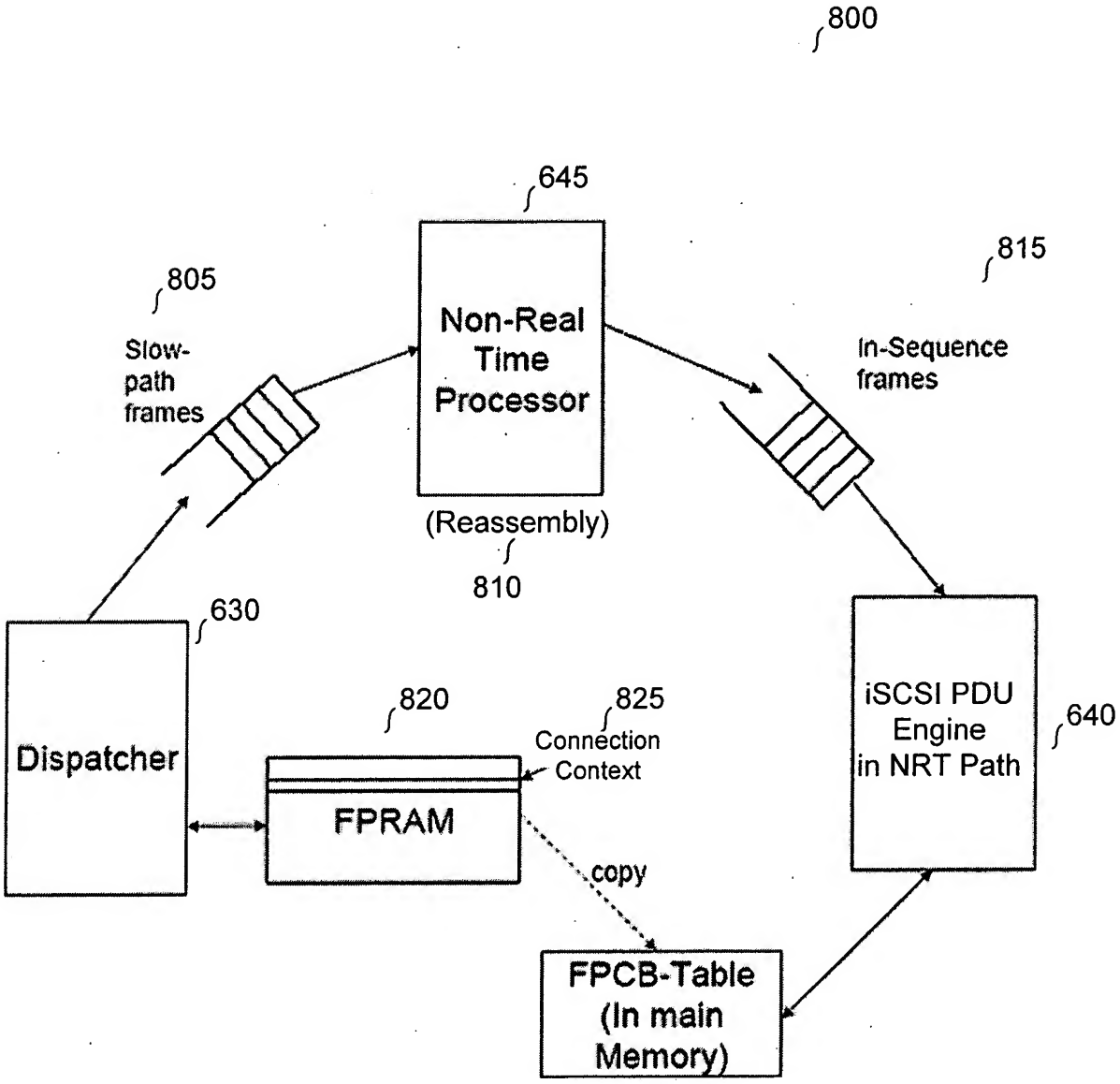

800
645
Non-Real Time Processor
(Reassembly)
810
805
Slow-path frames
815
In-Sequence frames
630
Dispatcher
820
FPRAM
825
Connection Context
copy
iSCSI PDU Engine in NRT Path
640
FPCB-Table (In main Memory)

FIGURE 17

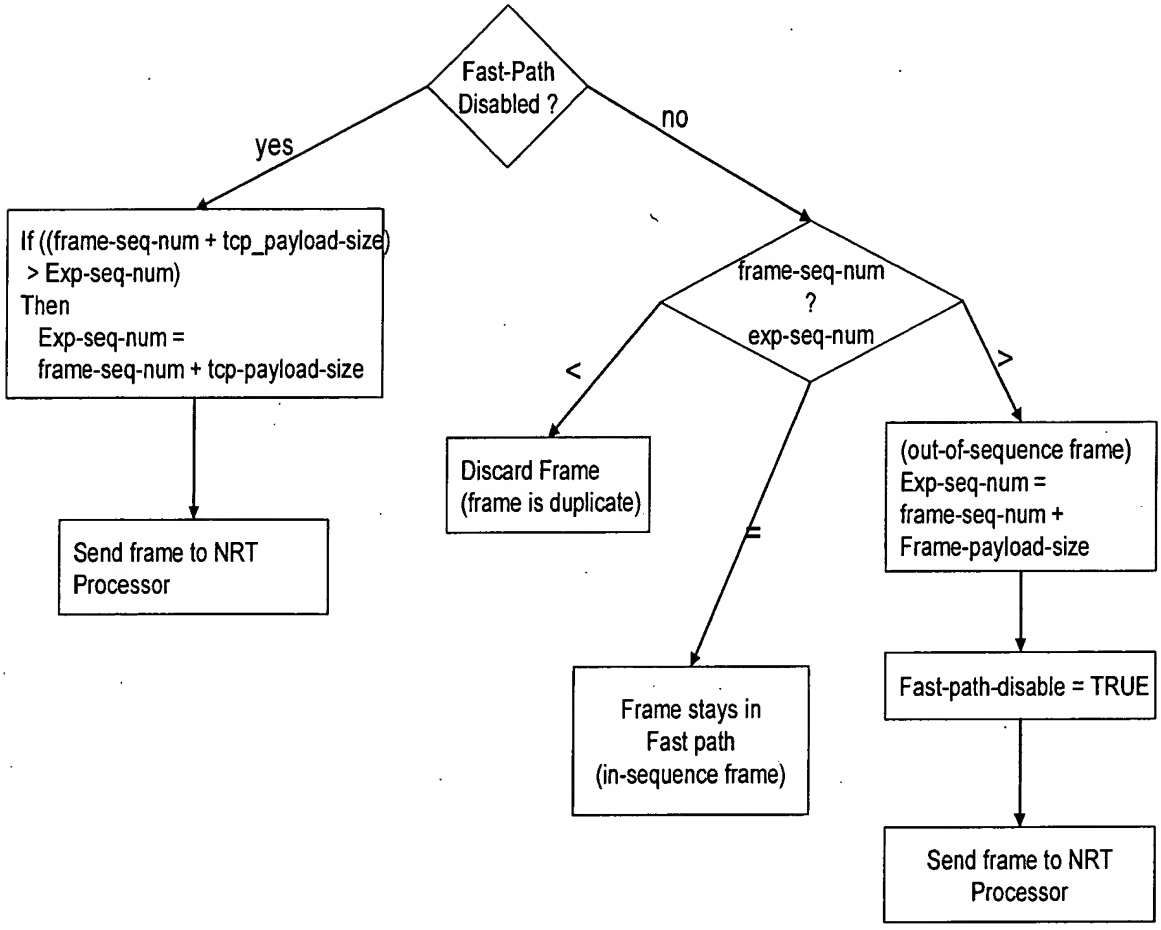

Fast-Path Disabled ?
yes
no
If ((frame-seq-num + tcp_payload-size) > Exp-seq-num) Then Exp-seq-num = frame-seq-num + tcp-payload-size
Send frame to NRT Processor
frame-seq-num ? exp-seq-num
<
=
>
Discard Frame (frame is duplicate)
Frame stays in Fast path (in-sequence frame)
(out-of-sequence frame) Exp-seq-num = frame-seq-num + Frame-payload-size
Fast-path-disable = TRUE
Send frame to NRT Processor

FIGURE 18A

byte#	#bits	field	Description
0	8		Control bits:
		sp	7: slow path forever (disallows automatic return to fast path)
		lo_pri	6: low priority (0=high pri). If set and awr_prx_rdy_lo false, discard frame
		ts_val	5: TCP timestamp valid. If invalid, timestamp check automatically passes
		rx_dis	4: PDU Rx disable. Do not store subsequent data or header segments
		flush	3: PDU flush. Do not store data segment, auto-reset flush at PDU end
		ddig_enb	2: data digest enable. Enables check of iSCSI data segment CRC
		hdig_enb	1: header digest enable. Enables check of iSCSI header segment CRC
		fp_enb	0: fastpath enable. If disabled, entire frame stored to Rx chunk.
1-4	32	nxt_seq	Next TCP sequence number expected
5	8	ts	TCP timestamp [17:10]
6-9	32	pcrc	iSCSI partial digest (checked so far)
10-12	24	dsctl	iSCSI data segment control:
	(1)	wenb	23: write SCSI data to buffer memory, vs header/non-SCSI data to Rx queue (flags how to interpret bytes 17-31)
	(23)	dsctr	22-0: data segment down counter (# words remaining in data seg, including data digest if present)
13-16	32	wptr	SCSI data write pointer ([3:0] indicates #qword residual bytes)
17-31	120		Within iSCSI header segments:
(17)	(8)	hctl	[7-4]=spare, [3]=final PDU, [2]=scsi data, [1:0]=#residual bytes
(18)	(8)	state	[7-5]=spare, [4:0]= state
(19-21)	(24)	wres	Word residual (up to 3 bytes)
(22)	(8)	ahctr	Additional header segment down counter (#words in AHSs)
(23-26)	(32)	hoffst	Data offset from data out or data in header
(27-28)	(16)	dplen	Data length from DPT
(29-31)	(24)	dpoffst	Data offset from DPT, bits 31:24 (sb 512B boundary)
			Within SCSI data segments:
17-31	120	qres	Qword residual (up to 15 bytes)

FIGURE 18B

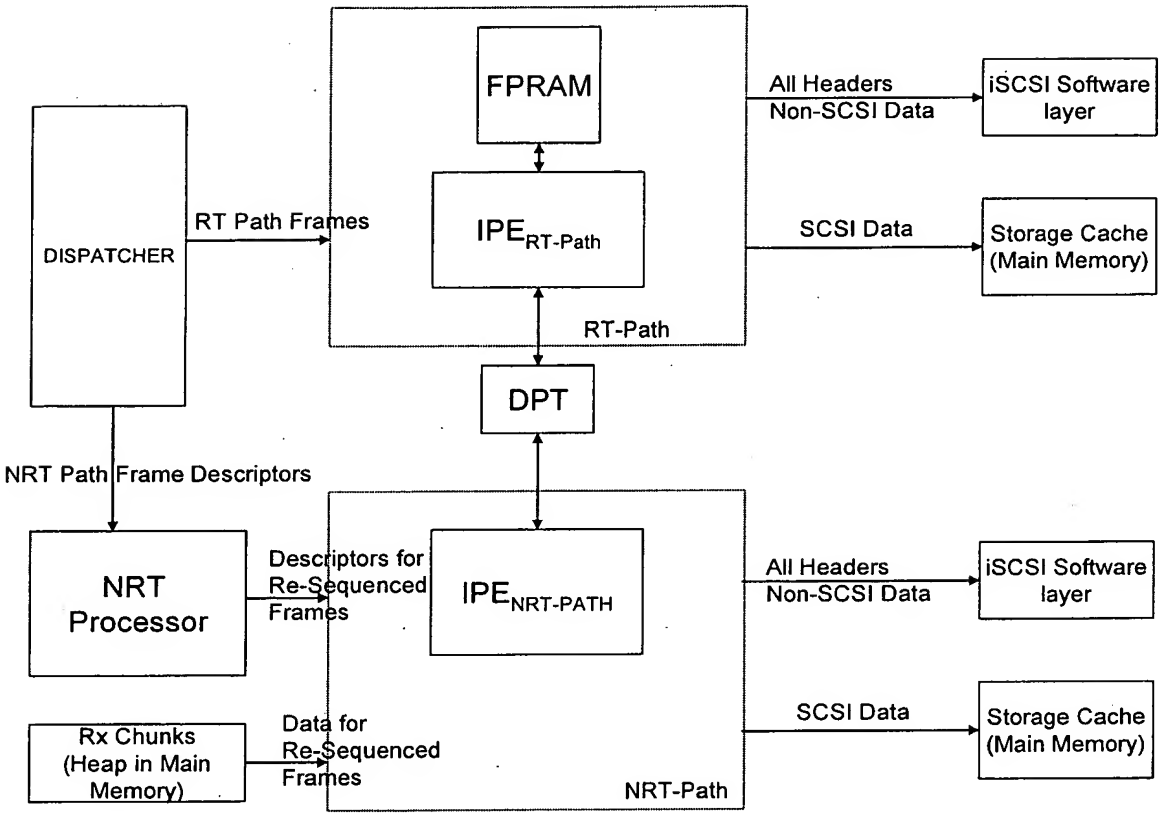

FPRAM
All Headers
Non-SCSI Data
iSCSI Software layer
DISPATCHER
RT Path Frames
IPE RT-Path
SCSI Data
Storage Cache (Main Memory)
RT-Path
DPT
NRT Path Frame Descriptors
NRT Processor
Descriptors for Re-Sequenced Frames
IPE NRT-PATH
All Headers
Non-SCSI Data
iSCSI Software layer
Rx Chunks (Heap in Main Memory)
Data for Re-Sequenced Frames
SCSI Data
Storage Cache (Main Memory)
NRT-Path

FIGURE 18C

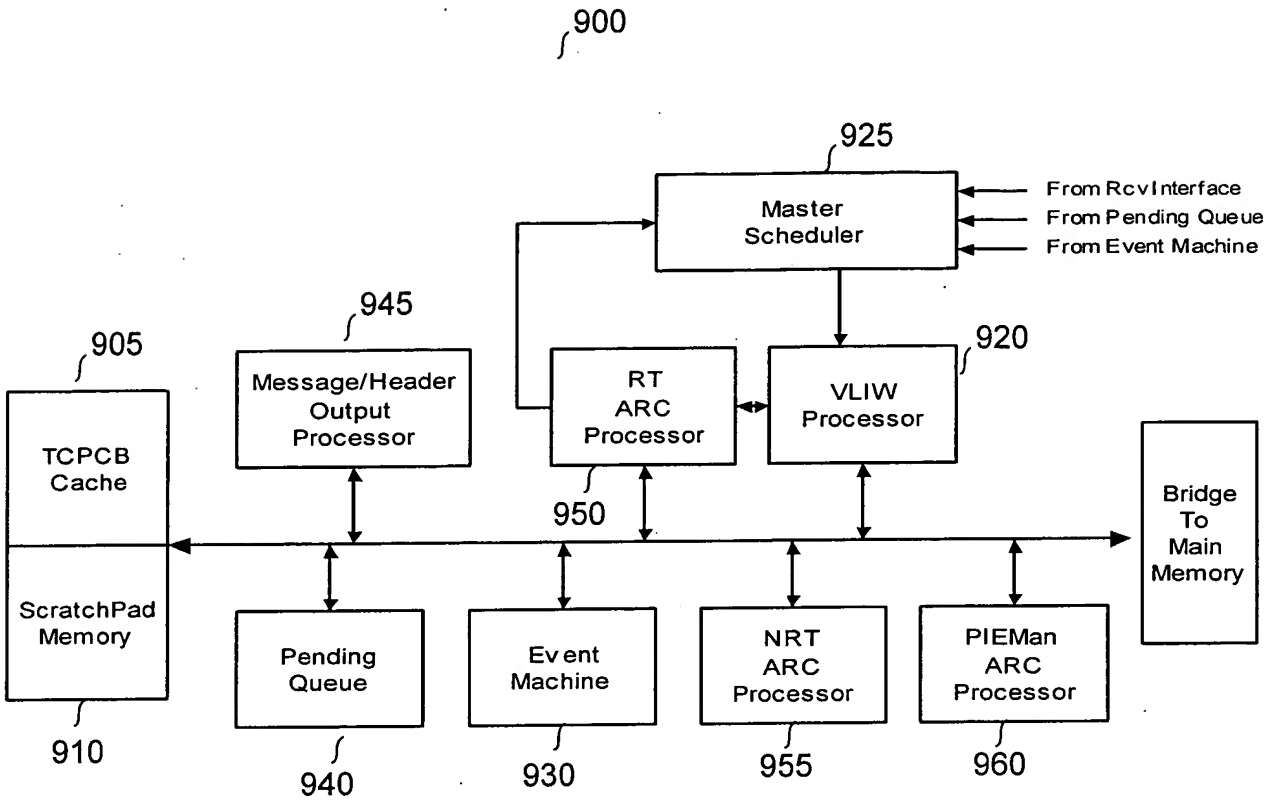

900
925
Master Scheduler
From Rcv Interface
From Pending Queue
From Event Machine
945
905
Message/Header Output Processor
RT ARC Processor
VLIW Processor
920
TCPCB Cache
950
Bridge To Main Memory
ScratchPad Memory
Pending Queue
Event Machine
NRT ARC Processor
PIEMan ARC Processor
910
940
930
955
960

FIGURE 19

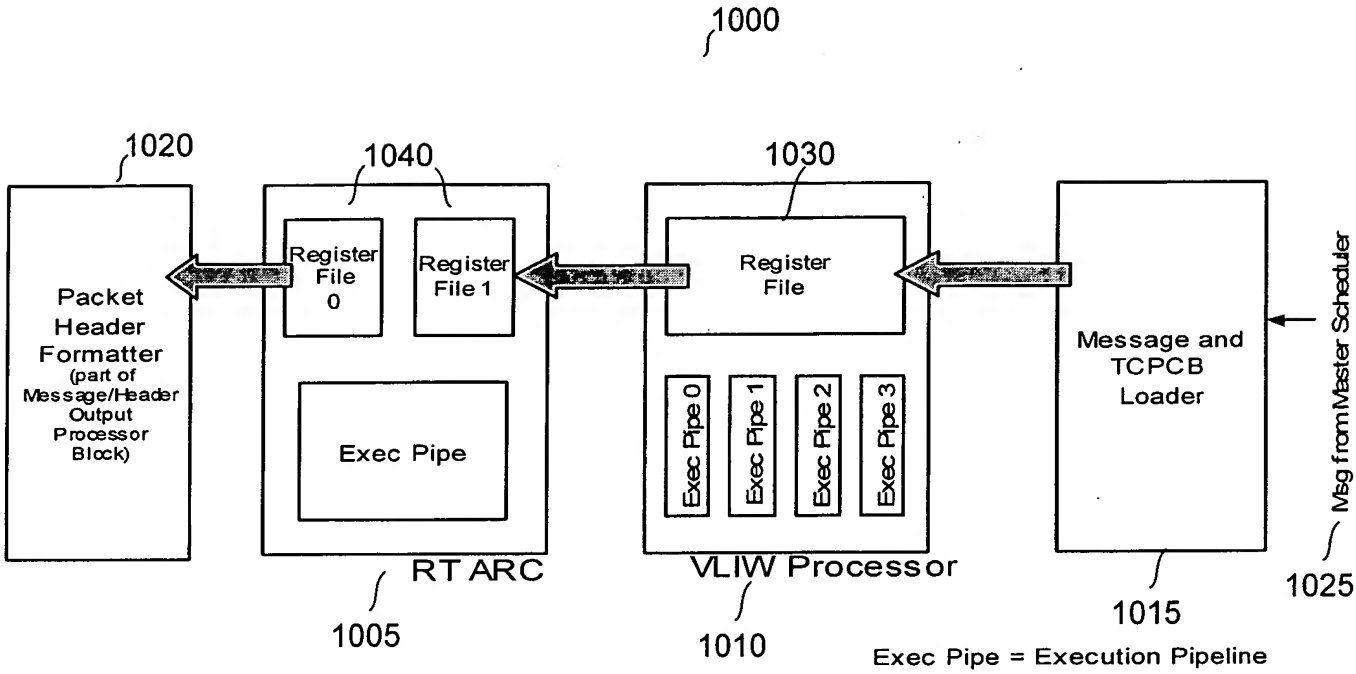

1000
1020
1040
1030
Packet Header Formatter (part of Message/Header Output Processor Block)
Register File 0
Register File 1
Exec Pipe
RT ARC
1005
Register File
Exec Pipe 0
Exec Pipe 1
Exec Pipe 2
Exec Pipe 3
VLIW Processor
1010
Message and TCPCB Loader
1015
Msg from Master Scheduler
1025
Exec Pipe = Execution Pipeline

FIGURE 20

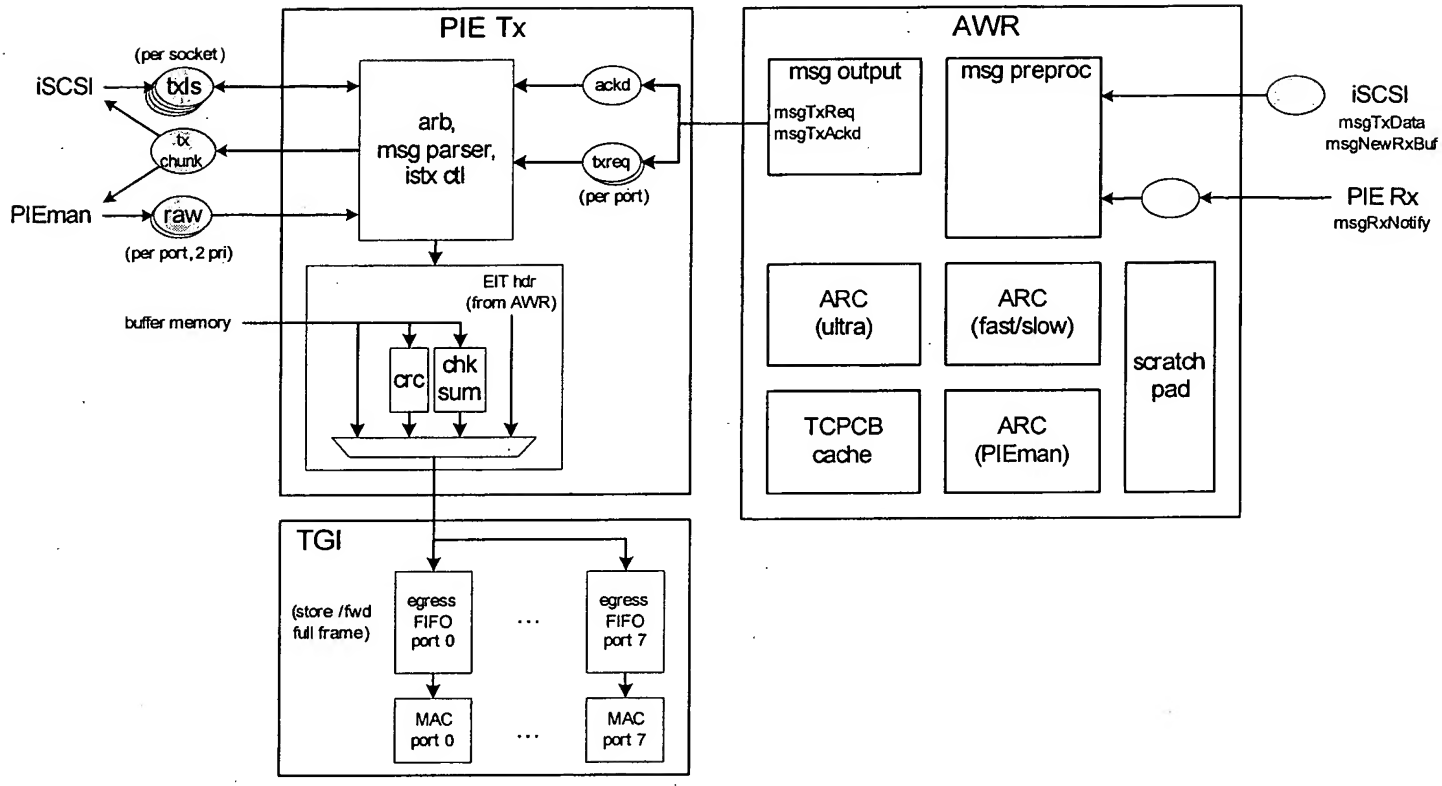

PIE Tx
AWR
(per socket)
iSCSI
txls
tx chunk
PIEman
raw
(per port, 2 pri)
arb, msg parser, istx ctl
ackd
txreq
(per port)
buffer memory
EIT hdr (from AWR)
crc
chk sum
msg output
msgTxReq
msgTxAckd
msg preproc
iSCSI
msgTxData
msgNewRxBuf
PIE Rx
msgRxNotify
ARC (ultra)
ARC (fast/slow)
scratch pad
TCPCB cache
ARC (PIEman)
TGI
(store /fwd full frame)
egress FIFO port 0
egress FIFO port 7
MAC port 0
MAC port 7

FIGURE 21

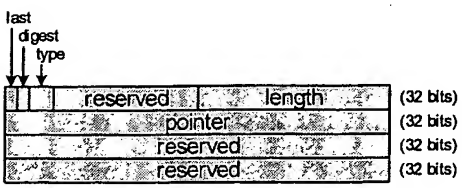


FIGURE 22

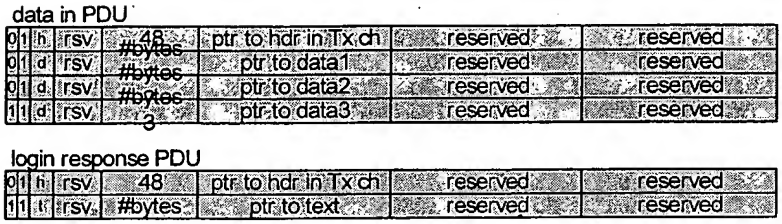


FIGURE 23

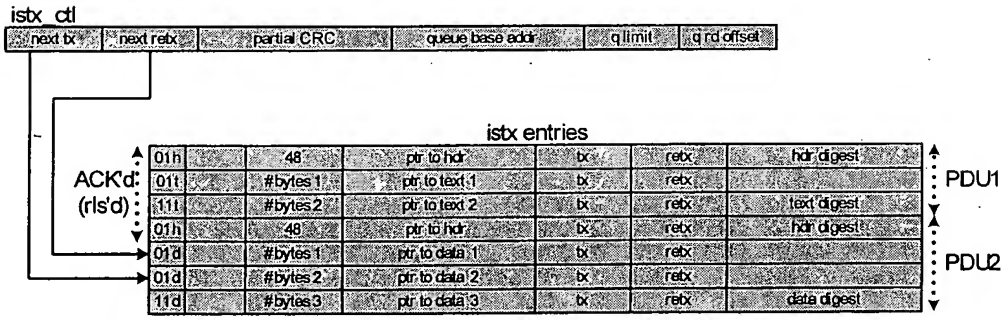


FIGURE 24

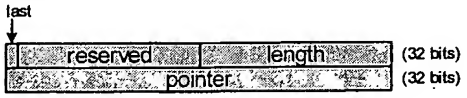


FIGURE 25

Command	Description
Push	Writes data beginning at write pointer and saves new write pointer in descriptor.
Push/Inc	Writes data beginning at write pointer, increments counter field by one, and saves new write pointer and counter in descriptor
Inc	Increments counter field by a specified amount and saves new counter in descriptor
Inc Bytes	Increments write pointer field by a specified amount and saves new write pointer in descriptor
Push/Chkpt	Writes data beginning at write pointer and saves new write pointer in descriptor and in descriptor extension as write checkpoint.
Push/Inc/Chkpt	Writes data beginning at write pointer, increments counter field by one, saves new write pointer and counter in descriptor, and saves new write pointer in descriptor extension as write checkpoint.
Inc/Chkpt	Increments counter field by a specified amount, saves new counter in descriptor, and copies current write pointer to write checkpoint
Rewind	Copies write checkpoint to write pointer and saves result in descriptor
Peek	Reads data beginning at read pointer (for queues) or write pointer (for stacks) but does not save new pointer in descriptor
Pop	Reads data beginning at read pointer (for queues) or write pointer (for stacks) and saves new pointer in descriptor
Pop/Dec	Reads data beginning at read pointer (for queues) or write pointer (for stacks), decrements counter field by one, and saves new pointer in descriptor
Dec	Decrements counter field by a specified amount and saves new counter in descriptor
Dec Bytes	Decrements read pointer (for queues) or write pointer (for stacks) by a specified amount and saves new pointer in descriptor

FIGURE 26

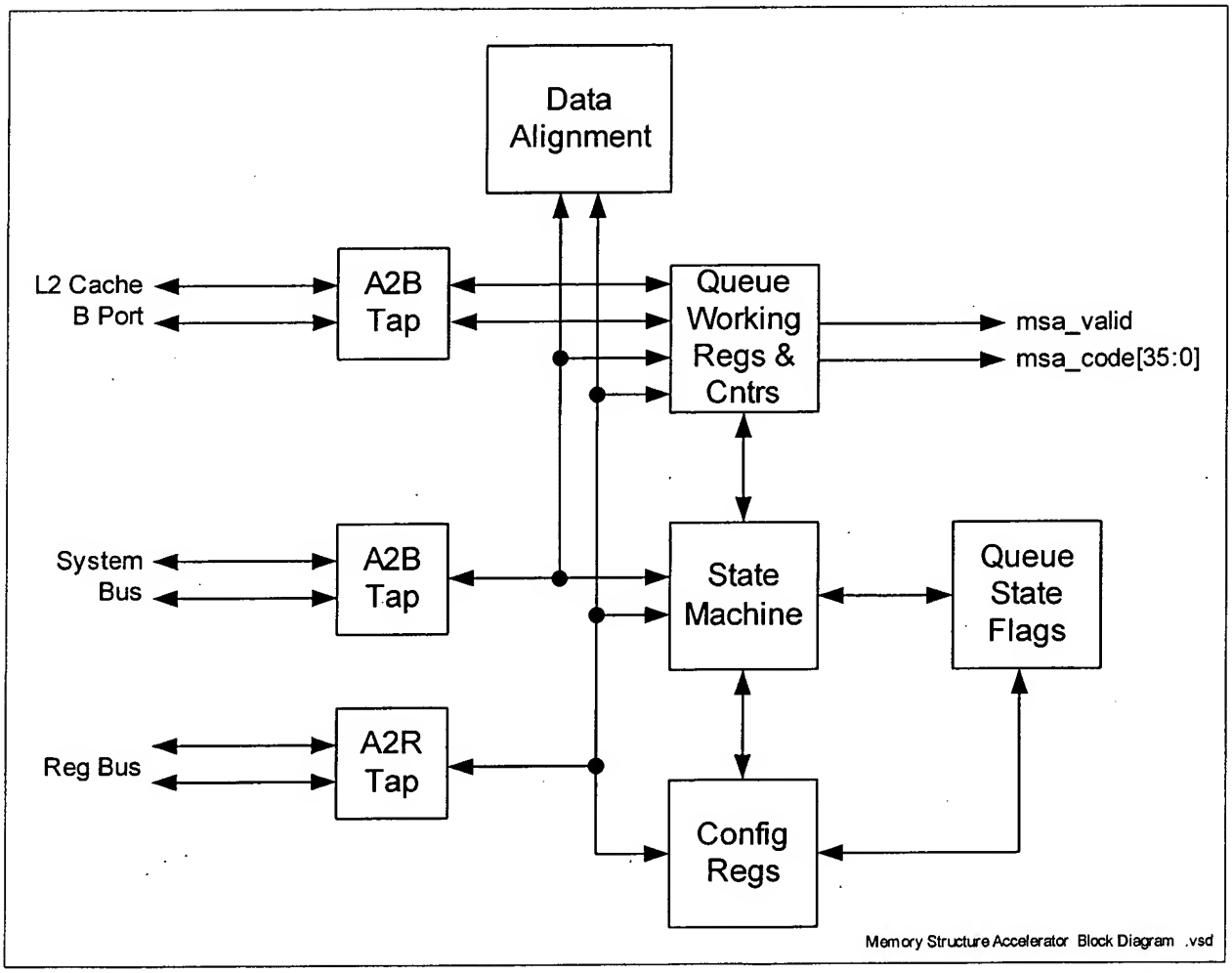

Data Alignment
L2 Cache B Port
A2B Tap
Queue Working Regs & Cntrs
msa_valid
msa_code[35:0]
System Bus
A2B Tap
State Machine
Queue State Flags
Reg Bus
A2R Tap
Config Regs
Memory Structure Accelerator Block Diagram .vsd

FIGURE 27

00000	- - -	01000	- - -	10000	- - -	11000	Rewind
00001	Pop/Dec (Read)	01001	Pop	10001		11001	Peek
00010	Push/Inc (Write)	01010	Push	10010	Push/Inc/Chkpt	11010	Push/Chkpt
00011	- - -	01011	Dec	10011	- - -	11011	- - -
00100	- - -	01100	- - -	10100	- - -	11100	- - -
00101	- - -	01101	- - -	10101	- - -	11101	- - -
00110	- - -	01110	Inc	10110	Inc/Chkpt	11110	- - -
00111	- - -	01111	- - -	10111	- - -	11111	- - -

FIGURE 28

00	Queue Not Empty	10	Queue Underflow
01	Queue Empty	11	Not used

FIGURE 29

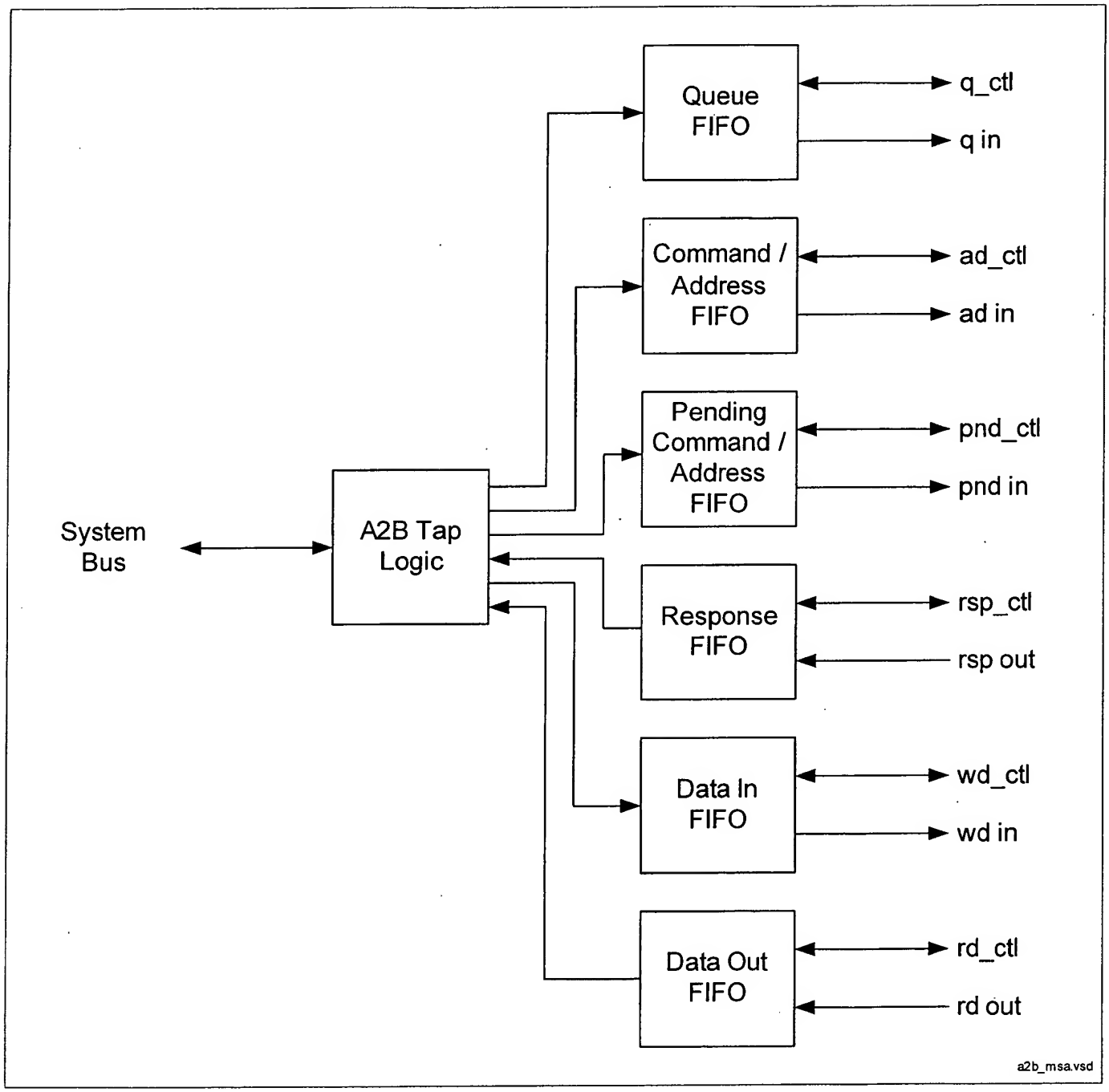

Queue FIFO
q_ctl
q in
Command / Address FIFO
ad_ctl
ad in
Pending Command / Address FIFO
pnd_ctl
pnd in
System Bus
A2B Tap Logic
Response FIFO
rsp_ctl
rsp out
Data In FIFO
wd_ctl
wd in
Data Out FIFO
rd_ctl
rd out
a2b_msa.vsd

FIGURE 30

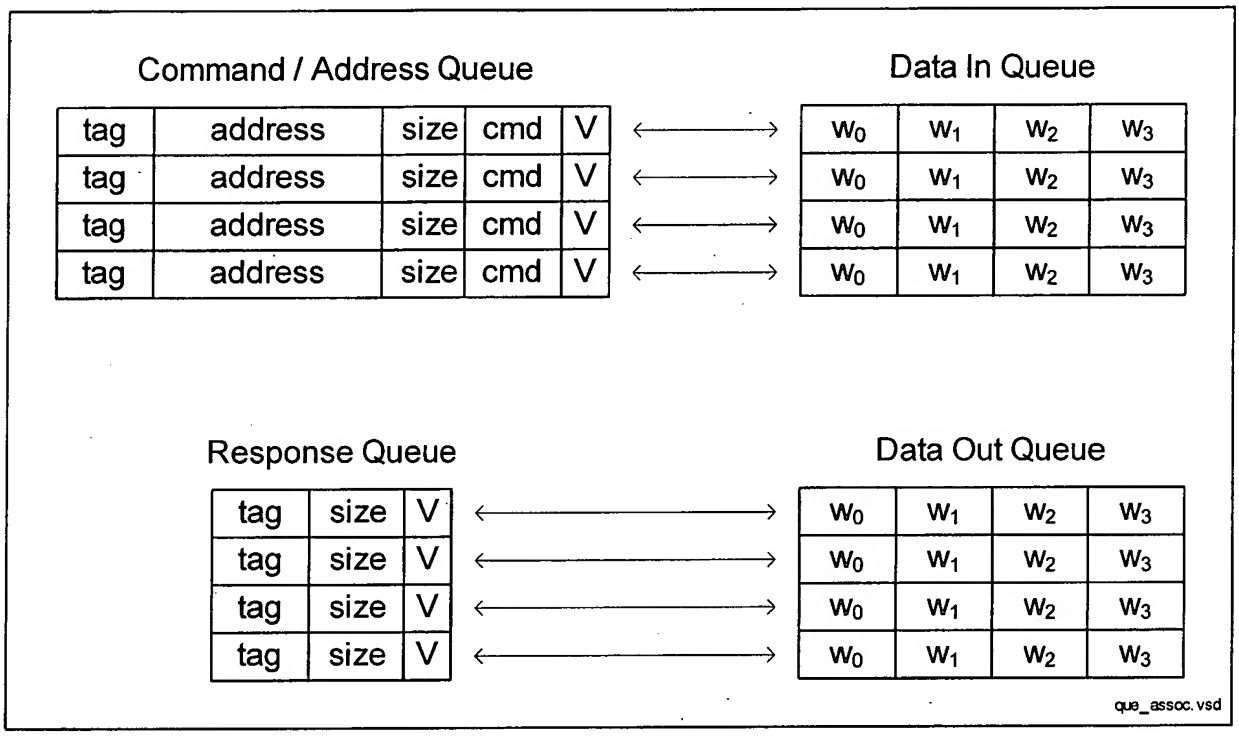
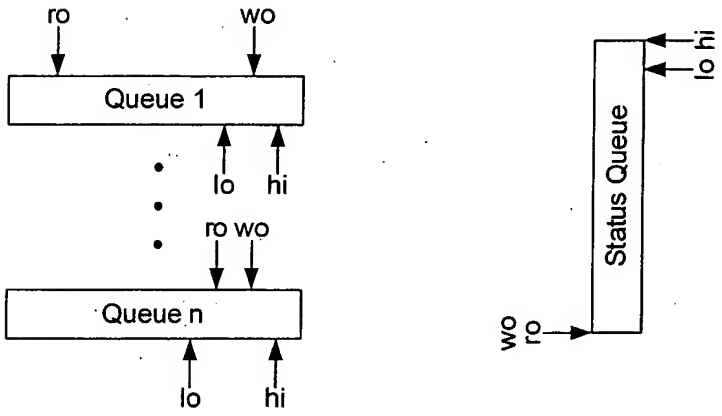
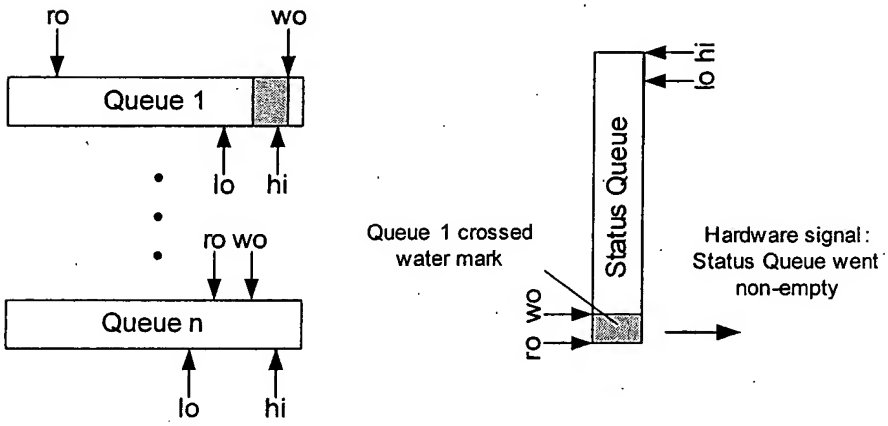

Command / Address Queue
tag address size cmd V
tag address size cmd V
tag address size cmd V
tag address size cmd V
Data In Queue
W0 W1 W2 W3
W0 W1 W2 W3
W0 W1 W2 W3
W0 W1 W2 W3
Response Queue
tag size V
tag size V
tag size V
tag size V
Data Out Queue
W0 W1 W2 W3
W0 W1 W2 W3
W0 W1 W2 W3
W0 W1 W2 W3
que_assoc.vsd

FIGURE 31

Current State



Push to Queue 1



Pop from Queue n

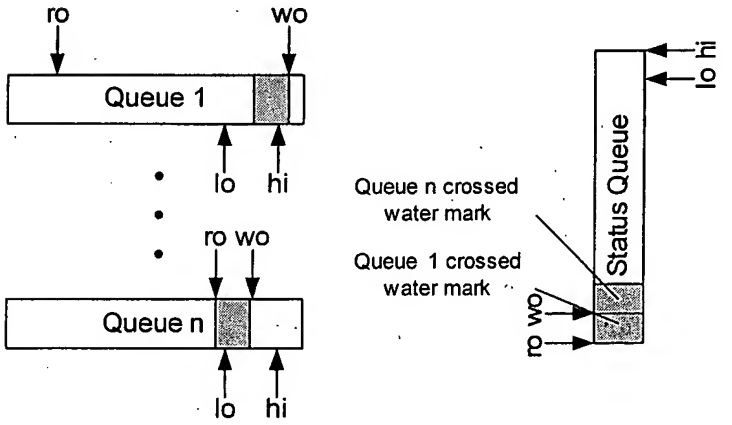


FIGURE 32

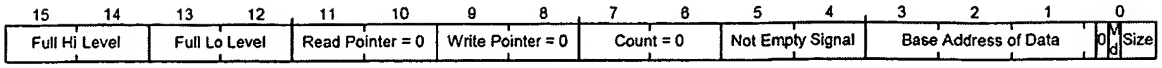


FIGURE 33

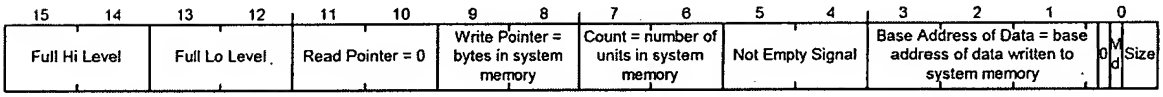


FIGURE 34